Программная обработка голосовых вызовов на мобильных телефонах

Асиненко А.А., Намиот Д.Е.

***Аннотация*— С ростом возможностей современных мобильных телефонов, они находят все новые и новые применения. Вместе с тем, есть некоторые разделы, практически не освещенные в работах и не получившие пока развития. В частности, к таким вопросам относится программное управление вызовами (call control) непосредственно на телефоне. Удивительно, но при всех возможностях и программных API современных смартфонов, например, отсутствуют (не описаны) возможности программно принимать и обрабатывать голосовые вызовы. То есть, собственно говоря, не описан программный способ использования телефон по прямому назначению. Именно такого рода системам и посвящена данная работа.**

***Ключевые слова*—телефония, call control API, Java ME, обработка вызовов, сервисы.**

I. Введение

С ростом возможностей современных мобильных телефонов, они находят все новые и новые применения. Вместе с тем, есть некоторые разделы, практически не освещенные в работах и не получившие пока развития. В частности, к таким вопросам относится программное управление вызовами (call control) непосредственно на телефоне. Удивительно, но при всех возможностях и программных API современных смартфонов, например, отсутствуют (не описаны) возможности программно принимать и обрабатывать голосовые вызовы. То есть, собственно говоря, не описан программный способ использования телефон по прямому назначению. Именно такого рода системам и посвящена данная работа.

Области применения очевидны: запуск (реализация) традиционных телекоммуникационных сервисов непосредственно на собственном мобильном телефоне (возможно – в связке с персональным компьютером). Мы имеем в виду такие сервисы, например, как автоответчик, переадресация, IVR и т.п.

В основу статьи положена квалификационная работа, выполненная на факультете ВМК МГУ им. М.В. Ломоносова.

Статья получена 10 августа 2013.

А.А. Асиненко – выпускник магистратуры ВМК МГУ

Д. Е. Намиот – старший научный сотрудник лаборатории ОИТ факультета ВМК МГУ имени М.В. Ломоносова.

II. The Java Telephony API(JSR-43)

Java Telephony API (JTAPI) [1] спецификация была разработана консорциумом ведущих в отрасли компьютерных и телекоммуникационных компаний, заинтересованных в разработке портативных, объектно-ориентированных API для компьютерной телефонии интегрированной управления вызовами.

Требования к этому API в том, чтобы просто и легко реализовать управление возможностями платформы CTI. Начальная рабочая группа состояла из представителей Intel ,Avaya , NortelNetworks , Novell и Sun Microsystems.

Но это API создано не для управления мобильными телефонами, поэтому в данном проекте данное API не применяется. А применяется Mobile Telephony API [2].

III. Общая схема работы

Для реализации проекта требуется мобильный телефон и компьютер. Телефон соединяется с компьютером через Wi-Fi, Bluetooth или data-кабель(только для общения с модемом). Чтобы реализовать возможности создания приложения-шлюза, которое будет перенаправлять вызовы в сеть интернет, требуется, также, соединение с интернетом.

Сама система – это два приложения, созданные с помощью Java ME и Java SE. Приложения взаимодействуют с собой по Wi-Fi или Bluetooth через сетевые протоколы TCP и UDP. TCP протокол используется для передачи команд управления, а UDP – для передачи аудиоданных.

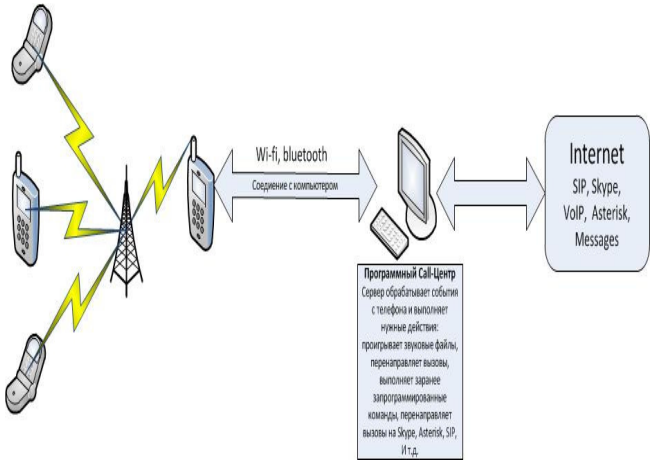

Рисунок 1. Схема проекта

Для передачи данных между телефоном и персональным компьютером создано клиент-серверное приложение.

Для управления вызовами телефона в Java ME существует несколько решений. Чтобы управлять вызовами была разработана спецификация JSR-253 (Mobile Telephony API), в разработке которой принимали участие такие компании как Motorola, Nokia, Sun Microsystems, Oracle и другие.
Спецификация Mobile Telephony API (MTA) определяет набор функций для управления голосовыми вызовами и управлением сети в Java приложениях, написанные для Java ME устройств. Например, Java приложения, использующие MTA, могут управлять конференцсвязью, сервисом запланированных звонков или службой голосового вызова. MTA оптимизирован для небольших устройств с ограниченным объемом памяти и вычислительной мощностью.
MTA не предназначен для всех телефонных сетей. Дизайн разработан расширяемым, так что уникальные функции, доступные в некоторых сетях могут распространяться в качестве дополнительных пакетов. MTA предоставляет услуги таким образом, что приложения могут использовать возможности телефонии, которые присутствуют на устройствах. MTA реализует следующие функции телефона:

- Инициирование голосовых и экстренных вызовов.
- Принятие входящих вызовов.
- Контроль и завершение текущего вызова.
- Получение уведомления о событиях вызова изменения состояния.
- Получение уведомления о событиях в сети изменения состояния (например, роуминга для различных сетей).
- Доступ к сетевой информации, такой как Network ID и доступным протоколам.
- Использование дополнительных услуг, таких как «многопартийные» звонки и переадресация вызова.
- Получение информацию о статусе дополнительных услуг.
Включение / выключение дополнительных услуг

Но сам API для Java ME программно реализован не был, и существует только в виде спецификации.
Основным способом управления вызовами телефона является управление модемом телефона. Модемом можно управлять как с приложения на компьютере через кабель передачи данных, так и из приложения работающем на самом телефоне (мидлета).
Подключение к модему происходит через COM-порт (последовательный порт). Модемом можно управлять только с помощью AT команд (HEYS команд).
Управлять вызовами телефона можно, только если есть возможность управлять модемом телефона. Естественно, что для этого не обходимо всего лишь подключение к COM-порту. К COM-порту можно подсоединиться как с приложения на компьютере, так и из Java приложения на телефоне (мидлета). После установки соединения, модем управляется с помощью AT команд.

Дальнейшие действия достаточно стандартны. Для того, чтобы модем распознал команды, они должны быть записаны в специфической форме. Каждая команда всегда начинается буквами AT или at (от англ. ATtention), а в конце завершается символом перевода строки. Команды воспринимаются модемом только тогда, когда он находится в "командном режиме" или offline. У каждой компании своя спецификация для управления модемом, но все основные принципы работы и команды одинаковы. Основные команды, которые используются для управления телефоном:
- ATD - позвонить
- ATA - ответить
- ATH – разорвать связь
- ! – переадресация телефонного звонка
- +++ - escape-команда
- / - повторить последнюю команду

Модем, при возникновении событий возвращает коды результатов, например:

- OK – модем успешно выполнил команду
- CONNECT – произошло соединение
- RING – возвращает при входящем вызове
- ERROR – возвращает если допущена ошибка в AT-команде
- BUSY – возвращает результат если занято

Чтобы управлять модемом из приложения на компьютере нужно подключить телефон с помощью data-кабеля к компьютеру, затем через COM-порт подключится к модему, перевести его в командный режим и далее посылать команды для управления вызовами. Для работы с COM-портом из приложений используется Java Communication API. Этот API можно использовать как для программирования последовательного порта(RS232/434, COM, tty), так и для программирования параллельного порта(принтер ,LPT). В Java Communication API есть абстрактный класс *CommPort* и два его подкласса: *SerialPort* и *ParallelPort*, которые описывают взаимодействие с последовательным и параллельным портом. Для работы с последовательным портом(COM) нужен класс SerialPort. Для получения данных с модема и отправки на него команд используются два метода: getInputStream() и getOutputStream(), которые работают с *java.io.InputStream* и *java.io.OutputStream*.

Чтобы управлять модемом из приложения на мобильном телефоне, также нужно подключиться к COM-порту. Для работы с COM-портом используется Generic Connection Framework.

Для работы с сетью в J2ME используется специальный API, входящий в MIDP: *Generic Connection Framework* (GCF). Поддержка GCF осуществляется на уровне конфигурации. Этот набор интерфейсов расположен в пакете *javax.microedition.io*.

Основу GCF составляет класс Connector и набор интерфейсов. Класс Connector является основой для образцов подключения. Интерфейсы определяют типы поддерживаемых соединений. Но этот API используется и для создания соединения с COM-портом. Ниже показан пример соединения.

```
(CommConnection)Connector.open("comm:COM0;baudrate
=19200", Connector.READ_WRITE, false);
```

После установки соединения общение между приложением и модемом, к которому мы обратились, происходит с помощью AT команд.

Исходящий телефонный вызов может быть сделан из приложений Java ME с помощью *javax.microedition.midlet.MIDlet.platformRequest()*. В этот метод нужно передать номер телефона в виде строки. Но с помощью этого метода можно только позвонить.

```
String telNo = “tel: +789191112233”;
platformRequest(telNo);
resumeRequest();
```

IV. Управление мультимедийными средствами телефона

Чтобы разработать персональный call-центр, не достаточно управлять только вызовами телефона. Нужно управлять мультимедийными возможностями телефона: захват звука из сети оператора, и воспроизведение звука в сеть. Для этого существует Mobile Media API (JSR-135) [3].

Для упрощения работы с существующими мобильными форматами существует Mobile Media API. API поддерживает любой основанный на времени аудио и видео контент, предлагая инструменты для управления мультимедийным потоком. API используется для захвата и воспроизведения аудиоданных.

Классы MMAPI находятся в пакете *javax.microedition.media*.

Класс Manager служит для создания плееров. Все плееры имеют интерфейс Player. В составе MMAPI есть и другие классы и интерфейсы, которые служат для управления громкостью, реакции на события и т. д.

Пустой интерфейс Control служит базой для конструирования различных интерфейсов управления проигрыванием. Несколько наследников Control определено в пакете *javax.microedition.media.control*: ToneControl, VolumeControl, MIDIControl и т. д.

Произвести захват звука можно следующим способом

```
Player p =
Manager.createPlayer(“capture://audio?encoding=amr”);
p.start()
```

V. Реализация

Для реализации поставленной задачи был разработан API. API разработан для платформы Java ME и использует COM порт мобильного телефона. На рисунке 2 показана общая диаграмма классов разработанного API.

В качестве базового пакета был создан пакет *javax.microedition.telcore*. В пакете *javax.microedition.telcore* находится основная функциональность API. Основным интерфейсом этого пакета является интерфейс TelConnection. Через этот интерфейс взаимодействуют все классы, которым нужно использовать соединения с COM портом или сетью. Этот интерфейс реализуют класс TelCOMConnection и интерфейс TelNetConnection, который, в свою очередь, реализуют два класса: TelUDPConnection и TelTCPConnection.

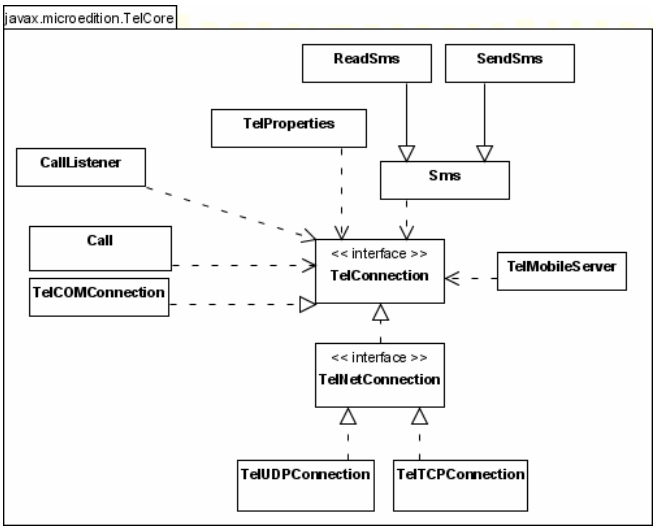

Рисунок 2. Диаграмма классов

Интерфейс TelConnection определяет основные методы и принципы создания и управления соединениями(COM, TCP, UDP). Класс TelCOMConnection реализует интерфейс TelConnection и служит для создания и управления соединения с COM портом. Это основной класс, потому что все управление телефоном происходит за счет управления модемом телефона через COM порт.

Для управления SMS сообщениями используется класс Sms. В классе Sms определены все способы управления сообщениями. Также предусмотрена рассылка сообщений (отправка сообщений на несколько номеров).

Класс TelProperties служит для настройки модема телефона.

Для того чтобы совершать вызовы, создан класс Call. В этом классе определены все методы, которые нужны для инициализации вызовов. Любое управление вызовами проходит в этом классе. CallListener – это класс для прослушивания COM-порта. Этот класс служит для обнаружения событий телефона и перехватывает все события модема.

Класс TelMobileServer создан для простой работы с сетью использует классы TelUDPConnection и TelTCPConnection. TelUDPConnection служит для управления дейтаграммами, которые служат для

передачи аудио данных. Для управления телефоном с компьютера используется TCP соединение.

На рисунке 3 показано взаимодействие классов Call, TelProperties и CallListener с классом TelCOMConnection через интерфейс TelConnection. Класс TelCOMConnection реализует все методы интерфейса TelConnection.в использовании для конечного пользователя.

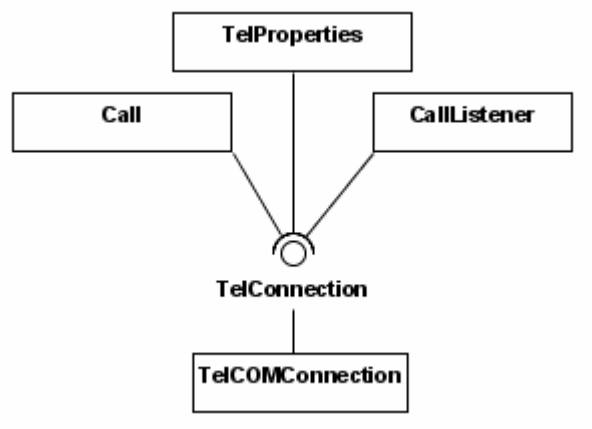

Рисунок 3. Взаимодействие классов управления вызовами

Говоря об использовании программной обработки голосовых вызовов, мы, в первую очередь, имеем в виду различные телекоммуникационные мэшапы [4]. Примеры подобного рода приложений можно найти в работах [5] и [6]. В задачах по автоматизации “умного дома” [7] голосовые вызовы использовались как один из возможных интерфейсов для пользователей.

БИБЛИОГРАФИЯ

[1] JSR-43: http://jcp.org/en/jsr/detail?id=43
[2] JSR-253 http://jcp.org/en/jsr/detail?id=253
[3] JSR-135: http://jcp.org/en/jsr/detail?id=135
[4] Schneps-Schneppe, M., & Namiot, D. (2008). Telco Enabled Social Networking: Russian Experience. In BALTIC CONFERENCE (p. 33).
[5] Schneps-Schneppe, M., & Namiot, D. (2008). Telco enabled social networking exampled by GEO tagging. APPLIED INFORMATION AND COMMUNICATION TECHNOLOGIES, 50.
[6] Schneps-Schneppe, M., & Namiot, D. (2013). About Home Gateway Mashups. International Journal of Open Information Technologies, 1(5), 1-5
[7] Schneps-Schneppe, M., Maximenko, A., Namiot, D., & Malov, D. (2012, October). Wired Smart Home: energy metering, security, and emergency issues. In Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on (pp. 405-410). IEEE. DOI: 10.1109/ICUMT.2012.6459700